METHOD OF PERFORMING A SIMON'S OR A SHOR'S QUANTUM ALGORITHM AND RELATIVE QUANTUM GATE

Field of the Invention

[0001] The present invention relates in general to quantum algorithms, and in particular, to a method and a corresponding quantum gate for performing Simon's and Shor's algorithms without the need for storing very large matrices.

Background of the Invention

[0002] Quantum algorithms are global random searching algorithms based on the principles, laws and quantum effects of quantum mechanics. They are used for controlling a process or for processing data in a database. They are particularly useful in performing intelligent processes of search-of-minima intelligent operations.

[0003] In quantum search operations, each design variable is represented by a finite linear superposition of classical initial states. Through a sequence of elementary unitary steps the initial quantum state |i> (for the input) is manipulated in such a way that a measurement of the final state of the system yields the correct output. Usually, in the quantum search algorithm three principal operators are used, namely: linear superposition (coherent states), entanglement, and interference.

[0004] For a better understanding of the field of

application of the invention, a brief description of quantum search algorithms is provided.

A SURVEY OF QUANTUM ALGORITHMS:

[0005] The problems solved by quantum algorithms may be stated as follows:

Input	A function $f:\{0,1\}^n \rightarrow \{0,1\}^m$
Problem	Find a certain property of f

The structure of a quantum algorithm is outlined, in a high level representation, by the diagram of FIGURE 1.

[0006] The input of a quantum algorithm is always a

BOX 1: UNITARY MATRIX U_F

A squared matrix U_F on the complex field is *unitary* if its inverse matrix coincides with its conjugate transpose:

$$U_F{}^{-1}=U_F{}^{\dagger}$$

A unitary matrix is always reversible and preserves the norm of vectors.

function f of a binary string into a binary string. This function is represented as a map table, defining for each string its image. The function f is first encoded into a unitary matrix operator U_F depending on the f properties. This operator calculates f when its input and output strings are encoded into canonical base vectors of a complex Hilbert space: U_F maps the vector code of every string into the vector code of its

image by f.

[0007] Once the matrix operator U_F has been generated, it is embedded into a quantum gate G. The quantum gate G is a unitary matrix whose structure depends on the form of the matrix U_F and on the problem to be solved. The quantum gate is the core of a quantum algorithm. In every quantum algorithm, the quantum gate acts on the initial canonical base of vectors (a same vector may always be chosen) to generate a complex linear combination (superposition) of base vectors as output. This superposition contains all the information to answer the initial problem.

[0008] After having made such a superposition, a measurement is done to extract this information. In quantum mechanics, a measurement is a non-deterministic operation that produces as output only one of the base vectors of the operation of superposition. The probability of each base vector being the output of the measurement depends on its complex coefficient (probability amplitude) of entering in the complex linear combination.

[0009] Each single operation of the quantum gate and the measurement form a quantum block. The quantum block is repeated k times to produce a collection of k base vectors. Since the measurement is a non-deterministic operation, these basic vectors will not necessarily be identical and each one of them encodes the information needed to solve the problem. The last part of the algorithm includes the interpretation of the collected base vectors to obtain with a certain probability the right answer to the initial problem.

ENCODER:

[00010] The behavior of the encoder block is described in the detailed schematic diagram of FIGURE 2. Function f is encoded into matrix U_F in three steps.

STEP 1:

[00011] The map table of function $f:\{0,1\}^n \rightarrow \{0,1\}^m$ is transformed into the map table of the injective function $F:\{0,1\}^{n+m} \rightarrow \{0,1\}^{n+m}$ such that:

$$F(x_0, \ldots, x_{n-1}, y_0, \ldots, y_{m-1}) =$$
$$(x_0, \ldots, x_{n-1}, f(x_0, \ldots, x_{n-1}) \oplus (y_0, \ldots, y_{m-1})) \quad (1)$$

BOX 2: XOR OPERATOR $\oplus$

The XOR operator between two binary strings p and q of length m is a string s of length m such that the i-th digit of s is calculated as the exclusive OR between the i-th digits of p and q:

$$p=(p_0, .., p_{n-1})$$
$$q=(q_0, .., q_{n-1})$$
$$s = p \oplus q = ((p_0+q_0) \bmod 2, .., (p_{n-1}+q_{n-1}) \bmod 2)$$

The need to deal with an injective function comes from the requirement that U_F be unitary. A unitary operator is reversible, so it cannot map two different inputs to the same output. Given that U_F is the matrix representation of F, F must be injective. Should the

matrix representation of function f be used, a non-unitary matrix may result, since f could be non-injective. Therefore, injectivity is assured by increasing the number of bits and by considering the function F instead of the function f. Anyway, function f can always be calculated from F by putting $(y_0, \ldots, y_{m-1})=(0, \ldots,0)$ in the input string and reading the last m values of the output string.

STEP 2:

[00012] The map table of function F is transformed into map table U_F, according to the following formula:

$$\forall s\in\{0,1\}^{n+m} : U_F[\tau(s)]= \tau [F(s)] \qquad (2)$$

The coding map $\tau :\{0,1\}^{n+m} \rightarrow \mathbf{C}^{2^{n+m}}$ ($\mathbf{C}^{2^{n+m}}$ is the target complex Hilbert space) is such that:

$$\tau(0)=\begin{pmatrix}1\\0\end{pmatrix}=|0\rangle \quad \tau(1)=\begin{pmatrix}0\\1\end{pmatrix}=|1\rangle$$
$$\tau(x_0,..,x_{n+m-1})=\tau(x_0)\otimes..\otimes\tau(x_{n+m-1})=|x_0..x_{n+m-1}\rangle \qquad (3)$$

BOX 3: VECTOR TENSOR PRODUCT ⊗

The tensor product between two vectors of dimensions h and k is a tensor product of dimension $h \cdot k$, such that:

$$|x\rangle \otimes |y\rangle = \begin{pmatrix} x_1 \\ \dots \\ x_h \end{pmatrix} \otimes \begin{pmatrix} y_1 \\ \dots \\ y_k \end{pmatrix} = \begin{pmatrix} x_1 y_1 \\ \dots \\ x_1 y_k \\ \dots \\ x_h y_1 \\ \dots \\ x_h y_k \end{pmatrix} \Rightarrow$$

Physical interpretation:

If a component of a complex vector is interpreted as the probability amplitude of a system being in a given state (indexed by the component number), the tensor product between two vectors describes the joint probability amplitude of two systems being in a joint state.

Examples: Vector Tensor Products

$$(0,0) \xrightarrow{\tau} \begin{pmatrix} 1 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} = |00\rangle$$

$$(0,1) \xrightarrow{\tau} \begin{pmatrix} 1 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} = |01\rangle$$

$$(1,0) \xrightarrow{\tau} \begin{pmatrix} 0 \\ 1 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} = |10\rangle$$

$$(1,1) \xrightarrow{\tau} \begin{pmatrix} 0 \\ 1 \end{pmatrix} \otimes \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} = |11\rangle$$

Code τ maps bit values into complex vectors of dimension 2 belonging to the canonical base of $\mathbf{C}^2$. Moreover, using tensor products, τ maps the general state of a binary string of dimension n into a vector of dimension 2^n, and reducing this state to the joint

state of the *n* bits forms the register. Every bit state is transformed into the corresponding 2-dimension base vector and then the string state is mapped into the corresponding 2^n-dimension base vector by composing all bit-vectors through tensor products. In this sense, a tensor product is the vector counterpart of state conjunction.

[00013] Base vectors are denoted using the *ket* notation $|i\rangle$. This notation is taken from Dirac's description of quantum mechanics.

STEP 3:

[00014] The map table of U_F is transformed into the matrix U_F by using the following transformation rule:

$$[U_F]_{ij} = 1 \Leftrightarrow U_F|j\rangle = |i\rangle \qquad (4)$$

which can be easily understood by considering the vectors $|i\rangle$ and $|j\rangle$ as column vectors. Because these vectors belong to the canonical base, U_F defines a permutation map of the rows of the identity matrix. In general, row $|j\rangle$ is mapped into row $|i\rangle$. This rule will be illustrated more in detail in a sample quantum algorithm, namely the Shor's algorithm.

QUANTUM BLOCK:

[00015] The core of the quantum block is the quantum gate, which depends on the properties of the matrix U_F. The scheme of FIGURE 3 gives a more detailed description of the quantum block. In FIGURE 3, the matrix operator U_F is the output of the encoder block represented in FIGURE 2. Here, it becomes the input for

the quantum block.

[00016] This matrix operator is first embedded into a more complex gate: the quantum gate G. Unitary matrix G is applied k times to an initial canonical base vector $|i\rangle$ of dimension 2^{n+m}. Every time, the resulting complex superposition $G|0..01..1\rangle$ of base vectors is measured, producing as a result one base vector $|x_i\rangle$. All the measured base vectors $\{|x_1\rangle,..,|x_k\rangle\}$ are collected together. This collection is the output of the quantum block.

[00017] The intelligence of such algorithms rests in the ability to build a quantum gate that is able to extract the information necessary to find the required property of f and to store it into the output vector collection. The structure of the quantum gate for every quantum algorithm will be discussed in detail, observing that a general description is possible. To represent quantum gates some special diagrams called quantum circuits will be used.

[00018] An example of a quantum circuit, relative to the so called Deutsch-Jozsa's quantum algorithm, is illustrated in FIGURE 4. Every rectangle is associated to a matrix $2^n \times 2^n$, where n is the number of lines entering and leaving the rectangle. For example, the rectangle marked U_F is associated to the matrix U_F. Typically, matrix H represents a Hadamard rotation:

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \qquad (5)$$

Quantum circuits provide a high-level description of the gate, and by using some transformation rules that

are listed in FIGURES 5a to 5f, it is possible to compile them into the corresponding gate-matrix.

BOX 4: MATRIX TENSOR PRODUCT ⊗

The tensor product between two matrices $X_{n\times m}$ and $Y_{h\times k}$ is a (block) matrix $(n\cdot h)\times(m\cdot k)$ such that:

$$X\otimes Y=\begin{bmatrix} x_{11}Y & .. & x_{1m}Y \\ .. & .. & .. \\ x_{n1}Y & .. & x_{nm}Y \end{bmatrix} \quad with \quad X=\begin{bmatrix} x_{11} & .. & x_{1m} \\ .. & .. & .. \\ x_{n1} & .. & x_{nm} \end{bmatrix}$$

Example: Matrix Tensor Product

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}\otimes\begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix}=\begin{bmatrix} 1\cdot\begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix} & 2\cdot\begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix} \\ 3\cdot\begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix} & 4\cdot\begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix} \end{bmatrix}=\begin{bmatrix} 5 & 6 & 10 & 12 \\ 7 & 8 & 14 & 16 \\ 15 & 18 & 20 & 24 \\ 21 & 24 & 28 & 32 \end{bmatrix}$$

[00019] The manner of using these rules will become clearer when a first sample of a quantum algorithm will be illustrated.

DECODER:

[00020] The decoder block has the function of interpreting the base vectors collected after the iterated execution of the quantum block. Decoding these vectors means to retranslate them back into binary strings and interpreting them directly if they already contain the answer to the starting problem. Alternatively, they may be used as coefficient vectors

for some system of equations to obtain the sought solution. This part will not be discussed in detail because it is relatively straightforward.

[00021] To better illustrate a field of application of the invention, a brief description of the Shor's algorithm is given.

SHOR'S ALGORITHM:

[00022] Shor's problem is so stated:

Given an integer number N, find a factor p for N.

[00023] This problem seems to be different from the other problems solved by quantum algorithms, but it can be reduced to an equivalent problem with the same form as the other quantum problems. This reduction is made possible by a result of the theory of numbers that relates the period *r* of a special periodic function to the factors of an integer *N*. This function is:

$$f_{N,a} : \aleph \rightarrow \aleph \; such\ that \; f_{N,a}(x) = a^x \bmod N$$

where *a* is a random number coprime to *N*, namely:

$$\gcd(a, N) = 1$$

where *gcd*(*x*, *y*) is the greatest common divisor between *x* and *y*.

[00024] This function is periodic (the period is at most *N*). Let the period be *r*. Then:

$$f_{N,a}(0) = f_{N,a}(r) \qquad a^r \equiv 1 \bmod N$$

If the period is even, this equation can be rewritten as:

$$\left(a^{r/2}\right)^2 \equiv 1 \bmod N \Leftrightarrow \left(a^{r/2}\right)^2 - 1 \equiv 0 \bmod N \Leftrightarrow \left(a^{r/2}-1\right)\left(a^{r/2}+1\right) \equiv 0 \bmod N$$

This means:

$$\exists h \in \aleph : \left(a^{r/2}-1\right)\left(a^{r/2}+1\right) = hN$$

So, unless $(a^{r/2}-1) \equiv 0 \bmod N$ or $(a^{r/2}+1) \equiv 0 \bmod N$, namely $a^{r/2} \equiv \pm 1 \bmod N$, at least one of $a^{r/2}+1$ or $a^{r/2}-1$ must have a non-trivial factor in common with N. It may be found by calculation:

$$gcd\left(a^{r/2}-1, N\right) \qquad gcd\left(a^{r/2}+1, N\right).$$

[00025] Using this reduction, the true question becomes: "What is the period of f?" Since the period of this function is less than N, it may be restricted to the interval [0, 1, ..., N-1]. Every input value is coded as a binary string. A number of bits n=[log N] (eventually [log N]+1) is needed to code all the N possible input values.

[00026] Therefore, Shor's problem is translated into the following standard quantum problem:

Input	$f:\{0,1\}^n \rightarrow \{0,1\}^n$ with period r
Problem	Find r

ENCODER:

[00027] First a simple example will be dealt with, then conclusions will be generalized.

A. Introductory example:

[00028] Consider the case:

$$N = 4 \Rightarrow n = 2; \qquad a = 3$$

Then, map table *f* is:

TABLE 1

(x_0, x_1)	$f(x_0, x_1)$
00	01
01	11
10	01
11	11

The period of this function is r=2.

STEP 1:

[00029] Function *f* is encoded into the injective function *F* built using eq. (2). Therefore, the map table of *F* is as follows:

TABLE 2

$(x_0, .., x_{n-1}, y_0, .., y_{n-1})$	$F(x_0, .., x_{n-1}, y_0, .., y_{n-1})$
0000	0001
0100	0111
1000	1001
1100	1111
0001	0000

0101	0110
1001	1000
1101	1110
0010	0011
0110	0101
1010	1011
1110	1101
0011	0010
0111	0100
1011	1010
1111	1100

STEP 2:

[00030] The function F is encoded into the map table of the operator U_F:

TABLE 3

$\|x_0 .. x_{n-1}\ y_0 .. y_{n-1}\rangle$	$U_F\|x_0 .. x_{n-1}\ y_0 .. y_{n-1}\rangle$
|0000>	|0001>
|0100>	|0111>
|1000>	|1001>
|1100>	|1111>
|0001>	|0000>
|0101>	|0110>
|1001>	|1000>
|1101>	|1110>
|0010>	|0011>
|0110>	|0101>
|1010>	|1011>
|1110>	|1101>
|0011>	|0010>
|0111>	|0100>

|1011>	|1010>
|1111>	|1100>

STEP 3:

[00031] The matrix corresponding to U_F is obtained by using the rule:

$$[U_F]_{ij} = 1 \Leftrightarrow U_F|j\rangle = |i\rangle$$

or in other words by observing that the first two vectors in the input tensor product are left unchanged, whereas the operator acting on the last two is chosen inside the set $\{I \otimes I,\ I \otimes C,\ C \otimes I,\ C \otimes C\}$ wherein I is the identity matrix and C is the complement matrix, depending on the values of the first two vectors:

TABLE 4

U_F	|00>	|01>	|10>	|11>
|00>	$I \otimes C$	0	0	0
|01>	0	$C \otimes C$	0	0
|10>	0	0	$I \otimes C$	0
|11>	0	0	0	$C \otimes C$

This matrix preserves the first two vectors and it preserves the third vector, flipping the fourth vector when the second vector is |0>; and flipping the third and fourth vectors when the second vector is |1>.

[00032] It is worth noting that the block matrix in cell (i, i) is identical to the block matrix in cell $((i+r) \bmod N, (i+r) \bmod N)$ where i is the binary label of the vector marking the matrix row and column of the cell.

B. Case with n=2:

[00033] In general, if n=2, a different value will be taken and so a different period r will be obtained, and the operator still maps the first n vectors into themselves but the block matrix pattern on the main diagonal changes.

[00034] The matrix U_F has the following form:

TABLE 5

U_F	|00>	|01>	|10>	|11>
|00>	M_{00}	0	0	0
|01>	0	M_{01}	0	0
|10>	0	0	M_{10}	0
|11>	0	0	0	M_{11}

where $M_i \in \{I \otimes I,\ I \otimes C,\ C \otimes I,\ C \otimes C\}$ and $M_i = M_j \Leftrightarrow ((j=i)$ OR $(j=(i+r) \bmod N))$.

C. General case:

[00035] Through a similar reasoning the following general form for the matrix U_F when $n>0$ may be proposed:

TABLE 6

U_F	|0..0>	|0..1>	...	|1..1>
|0..0>	$M_{0..0}$	0	...	0
|0..1>	0	$M_{0..1}$	...	0
...	...	...	...	...
|1..1>	0	0	0	$M_{1..1}$

where $M_i = P_1 \otimes .. \otimes P_n$, $P_k \in \{I,\ C\}$, $k=1,..,n$ and $M_i = M_j \Leftrightarrow ((j=i) \text{OR} (j=(i+r) \bmod N))$. The column labels are base vectors of dimension n.

QUANTUM BLOCK:

[00036] The Shor's quantum gate may be described with the quantum circuit of FIGURE 6. Shor's quantum gate shown in FIGURE 7 is very similar to Simon's quantum gate shown in FIGURE 8, unless for the last operator.

[00037] In Simon's algorithm the operator ${}^{n}H$ is applied to the first n vectors exiting from U_F. This operator of the Simon's algorithm is responsible for the deletion of some of the cells in the first column of the final gate and this produces a special superposition of base vectors as output. By measuring this superposition it is possible to solve the Simon's problem.

[00038] In Shor's algorithm the interference operator is not ${}^{n}H$, in fact the main diagonal pattern of U_F is different, and therefore a different interference operator is needed to extract the information. This operator is represented by the matrix QFT_n, called Quantum Fourier Transform of order n. This operator is non-classical because it maps a base vector into a complex linear combination of base vectors. In general, a base vector $|i\rangle$ is mapped into the linear combination $\alpha_1 y_1 + \ldots + \alpha_{2^n} y_{2^n}$ where α_i and α_{i+1} have the same absolute value $1/2^{n/2}$ but they are phase shifted $i \cdot (2\pi/2^n)$ starting with $\alpha_1 = 1/2^{n/2}$. The operator is defined as follows:

TABLE 7

QFT_n	$\phi=0$	$\phi=2\pi/2^n$	...	$\phi=(2^n-1)2\pi/2^n$
	|0..0>	|0..1>	...	|1..1>
|0..0>	$1/2^{n/2}$	$1/2^{n/2}$	...	$1/2^{n/2}$
|0..1>	$1/2^{n/2}$	$1/2^{n/2}\ e^{J2\pi/2^n}$	...	$1/2^{n/2}\ e^{J\ (2^n-1)\ 2\pi/2^n}$
...	...	...	...	...
|1..1>	$1/2^{n/2}$	$1/2^{n/2}\ e^{J\ (2^n-1)\ 2\pi/2^n}$	...	$1/2^{n/2}\ e^{J\ (2^n-1)^2\ 2\pi/2^n}$

where J is the imaginary unit. By using the transformation rules shown in FIGURES 5a to 5f to the circuit of FIGURE 6, the quantum gate of FIGURE 7 is obtained.

[00039] A discussion of the form of these gates for a few particular cases will follow and the observations will then be generalized.

A. Introductory example:

[00040] If n=2, the quantum gate has the following form:

$$G=(QFT_2 \otimes^2 I)\cdot U_F \cdot (^2H \otimes^2 I)$$

This gate is now calculated for the first considered example:

TABLE 8

$^2H\otimes^2I$	|00>	|01>	|10>	|11>
|00>	$^2I/2$	$^2I/2$	$^2I/2$	$^2I/2$
|01>	$^2I/2$	$-^2I/2$	$^2I/2$	$-^2I/2$
|10>	$^2I/2$	$^2I/2$	$-^2I/2$	$-^2I/2$
|11>	$^2I/2$	$-^2I/2$	$-^2I/2$	$^2I/2$

The matrix U_F has been already presented in TABLE 4, therefore

TABLE 9

U_F ($^2H \otimes ^2I$)	|00>	|01>	|10>	|11>
|00>	$I \otimes C/2$	$I \otimes C/2$	$I \otimes C/2$	$I \otimes C/2$
|01>	$C \otimes C/2$	$-C \otimes C/2$	$C \otimes C/2$	$-C \otimes C/2$
|10>	$I \otimes C/2$	$I \otimes C/2$	$-I \otimes C/2$	$-I \otimes C\ /2$
|11>	$C \otimes C/2$	$-C \otimes C/2$	$-C \otimes C/2$	$C \otimes C/2$

If $n = 2$, QFT_2 is as follows:

TABLE 10

QFT_2	$\phi=0$	$\phi=\pi/2$	$\phi=\pi$	$\phi=3\pi/2$
	|00>	|01>	|10>	|11>
|00>	1/2	1/2	1/2	1/2
|01>	1/2	J/2	-1/2	-J/2
|10>	1/2	-1/2	1/2	-1/2
|11>	1/2	-J/2	-1/2	J/2

TABLE 11

$QFT_2 \otimes ^2I$	|00>	|01>	|10>	|11>
|00>	$^2I/2$	$^2I/2$	$^2I/2$	$^2I/2$
|01>	$^2I/2$	J $^2I/2$	$-^2I/2$	-J $^2I/2$
|10>	$^2I/2$	$-^2I/2$	$^2I/2$	$-^2I/2$
|11>	$^2I/2$	-J $^2I/2$	$-^2I/2$	J $^2I/2$

The table defining the quantum gate G is calculated as follows:

TABLE 12

G	|00>	|01>	|10>	|11>
|00>	$(I\otimes C+C\otimes C)/2$	$(I\otimes C-C\otimes C)/2$	0	0
|01>	0	0	$(I\otimes C+JC\otimes C)/2$	$(I\otimes C-JC\otimes C)/2$
|10>	$(I\otimes C-C\otimes C)/2$	$(I\otimes C+C\otimes C)/2$	0	0
|11>	0	0	$(I\otimes C-JC\otimes C)/2$	$(I\otimes C+JC\otimes C)/2$

By applying the operator G to the vector |0000> the following is obtained:

$$G|0000\rangle = |00\rangle\frac{1}{2}(I\otimes C+C\otimes C)|00\rangle + |10\rangle\frac{1}{2}(I\otimes C-C\otimes C)|00\rangle .$$

[00041] If a measurement of this vector is made and the first two vectors of dimension 2 are encoded back into their binary labels, the possible results are:

00 with probability 0.5
10 with probability 0.5

The distance between these values is $d=[|10-00|]_{10}=[10]_{10}=2$, where $[s]_{10}$ is the decimal representation of the binary string s. It should be observed that $N/r=4/2=2$, therefore $d=N/r$. If r is unknown, it may be calculated as:

$$r=N/d.$$

B. Case with n=2, r=2:

[00042] As shown above, when $n=2$, the quantum gate has the following form:

$$G=(QFT_2\otimes^2 I)\cdot U_F\cdot(^2H\otimes^2 I)$$

by using the matrices calculated in the introductory example and recalling U_F given by TABLE 5.

TABLE 13

U_F ($^2H \otimes ^2I$)	|00>	|01>	|10>	|11>
|00>	$M_{00}/2$	$M_{00}/2$	$M_{00}/2$	$M_{00}/2$
|01>	$M_{01}/2$	$-M_{01}/2$	$M_{01}/2$	$-M_{01}/2$
|10>	$M_{10}/2$	$M_{10}/2$	$-M_{10}/2$	$-M_{10}/2$
|11>	$M_{11}/2$	$-M_{11}/2$	$-M_{11}/2$	$M_{11}/2$

[00043] By recalling the expression of the interference operator $QFT_2 \otimes ^2I$ shown in TABLE 11, the following generalized form of G is obtained:

TABLE 14

G	|00>	|01>	|10>	|11>
|00>	$(M_{00}+M_{01}+M_{10}+M_{11})/4$	$(M_{00}-M_{01}+M_{10}-M_{11})/4$	$(M_{00}+M_{01}-M_{10}-M_{11})/4$	$(M_{00}-M_{01}-M_{10}+M_{11})/4$
|01>	$(M_{00}+JM_{01}-M_{10}-JM_{11})/4$	$(M_{00}-JM_{01}-M_{10}+JM_{11})/4$	$(M_{00}+JM_{01}+M_{10}+JM_{11})/4$	$(M_{00}-JM_{01}+M_{10}-JM_{11})/4$
|10>	$(M_{00}-M_{01}+M_{10}-M_{11})/4$	$(M_{00}+M_{01}+M_{10}+M_{11})/4$	$(M_{00}-M_{01}-M_{10}+M_{11})/4$	$(M_{00}+M_{01}-M_{10}-M_{11})/4$
|11>	$(M_{00}-JM_{01}-M_{10}+JM_{11})/4$	$(M_{00}+JM_{01}-M_{10}-JM_{11})/4$	$(M_{00}-JM_{01}+M_{10}-JM_{11})/4$	$(M_{00}+JM_{01}+M_{10}+JM_{11})/4$

If r=2, like in our introductory example, then $M_{00}=M_{10} \neq M_{01}=M_{11}$. This means:

TABLE 15

G	|00>	|01>	|10>	|11>
|00>	$(M_{00}+M_{01})/2$	$(M_{00}-M_{01})/2$	0	0
|01>	0	0	$(M_{00}+JM_{01})/2$	$(M_{00}-JM_{01})/2$
|10>	$(M_{00}-M_{01})/2$	$(M_{00}+M_{01})/2$	0	0
|11>	0	0	$(M_{00}-JM_{01})/2$	$(M_{00}+JM_{01})/2$

Consider the application of *G* to vector |0000>:

$$G|0000\rangle = |00\rangle\frac{1}{2}(M_{00}+M_{01})|00\rangle + |10\rangle\frac{1}{2}(M_{00}-M_{01})|00\rangle \ .$$

[00044] If a measurement is done on this vector and the first two vectors of dimension 2 are encoded back into their binary labels, then the possible results are:

00 with probability 0.5
10 with probability 0.5

These are the same results that were obtained for our introductory example and the same conclusions hold.

C. General case:

[00045] As already shown, for a general case, the operator U_F is defined as in TABLE 6. The quantum gate G will now be calculated for a general situation:

TABLE 16

$^nH \otimes {}^nI$	$\|0..0\rangle$	...	$\|j\rangle$	...	$\|1..1\rangle$
$\|0..0\rangle$	$^nI/2^{n/2}$	...	$^nI/2^{n/2}$	...	$^nI/2^{n/2}$
$\|0..1\rangle$	$^nI/2^{n/2}$	...	$(-1)^{(0..1)\cdot j}$ $(^nI/2^{n/2})$	...	$-\,{}^nI/2^{n/2}$
...	...	...	...	...	...
$\|i\rangle$	$^nI/2^{n/2}$	...	$(-1)^{i\cdot j}$ $(^nI/2^{n/2})$	...	$(-1)^{i\cdot(1..1)}$ $(^nI/2^{n/2})$
...	...	...	...	...	...
$\|1..1\rangle$	$^nI/2^{n/2}$	...	$(-1)^{(1..1)\cdot j}$ $(^nI/2^{n/2})$	...	$(-1)^{(1..1)\cdot(1..1)}$ $(^nI/2^{n/2})$

TABLE 17

$U_F \cdot (^nH \otimes {}^nI)$	$\|0..0\rangle$	...	$\|j\rangle$	...	$\|1..1\rangle$
$\|0..0\rangle$	$M_{0..0}/2^{n/2}$	...	$M_{0..0}/2^{n/2}$	...	$M_{0..0}/2^{n/2}$
...	...	...	...	...	...
$\|i\rangle$	$M_i/2^{n/2}$	...	$(-1)^{i\cdot j}$ $M_i/2^{n/2}$	...	$(-1)^{i\cdot(1..1)}$ $M_i/2^{n/2}$
...	...	...	...	...	...
$\|1..1\rangle$	$M_{1..1}/2^{n/2}$	...	$(-1)^{(1..1)\cdot j}$ $M_{1..1}/2^{n/2}$	...	$(-1)^{(1..1)\cdot(1..1)}$ $M_{1..1}/2^{n/2}$

Observe that:

$$[QFT]_{i,j} = \frac{1}{2^{n/2}} e^{J[i]_{10}\frac{[j]_{10}\cdot 2\pi}{2^n}}$$

where $[i]_{10}$ and $[j]_{10}$ are the decimal representations of the binary strings i and j. Therefore:

TABLE 18

$QFT_n \otimes {}^nI$	$\|0..0\rangle$	...	$\|j\rangle$	...	$\|1..1\rangle$
$\|0..0\rangle$	${}^nI/2^{n/2}$	...	${}^nI/2^{n/2}$	...	${}^nI/2^{n/2}$
...	...	...	...	...	...
$\|i\rangle$	${}^nI/2^{n/2}$	...	${}^nI/2^{n/2} e^{J[i]_{10}\cdot[j]_{10} 2\pi/2^n}$	...	${}^nI/2^{n/2} e^{J[i]_{10}\cdot(2^n - 1) 2\pi/2^n}$
...	...	...	...	...	...
$\|1..1\rangle$	${}^nI/2^{n/2}$	...	${}^nI/2^{n/2} e^{J(2^n - 1)\cdot[j]_{10} 2\pi/2^n}$	...	${}^nI/2^{n/2} e^{J(2^n -1)^2 2\pi/2^n}$

The quantum gate G has the following form:

TABLE 19

G	$\|0..0\rangle$	...
$\|0..0\rangle$	$1/2^n \sum_{k\in\{0,1\}^n} e^{J\pi \cdot 0 \cdot [k]_{10}/2^{n-1})} M_k$	...
...	...	...
$\|i\rangle$	$1/2^n \sum_{k\in\{0,1\}^n} e^{J\pi \cdot [i]_{10} \cdot [k]_{10}/2^{n-1}} M_k$	...
...	...	...
$\|1..1\rangle$	$1/2^n \sum_{k\in\{0,1\}^n} e^{J\pi \cdot (2^n -1) \cdot [k]_{10}/2^{n-1}} M_k$	...

Consider the term:

$$1/2^n \sum_{k\in\{0,1\}^n} e^{J\pi \cdot [i]_{10} \cdot [k]_{10}/2^{n-1}} M_k$$

Since $M_i = P_1 \otimes .. \otimes P_n$, $P_k \in \{I, C\}$, $k=1,..,n$ and $M_i = M_j \Leftrightarrow$ $((j=i) \text{OR} (j=(i+r) \text{mod} N))$ this term may be written as:

$$1/2^n \sum_{h\in R} [e^{J\pi \cdot [i]_{10} \cdot [h]_{10}/2^{n-1}} + e^{J\pi \cdot [i]_{10} \cdot ([h]_{10}+1r)/2^{n-1}} + .. + e^{J\pi \cdot [i]_{10} \cdot ([h]_{10}+(l_k-1)r)/2^{n-1}}] M_h$$

or:

$$1/2^n\Sigma_{h\in R}\ (-1)^{[h]_{10}[i]_{10}/2^{n-1}}\ [(-1)^{0r[i]_{10}/2^{n-1}} + (-1)^{1r[i]_{10}/2^{n-1}} + ..$$
$$+ (-1)^{(l_h-1)r[i]_{10}/2^{n-1}}]M_h$$

where $R=\{0..0,0..1, .., [r-1]_2\}$. Suppose that N is a multiple of r, then $l_h=2^n/r=l$ for every h. Therefore, the previous term can be transformed into:

$$1/2^n\Sigma_{h\in R}\ (-1)^{[h]_{10}[i]_{10}/2^{n-1}}\ [(-1)^{2\cdot 0\cdot [i]_{10}/l} + (-1)^{2\cdot 1\cdot [i]_{10}/l} +$$
$$.. + (-1)^{2\cdot (l-1)\cdot [i]_{10}/l}]M_h$$

and finally:

$$1/2^n\Sigma_{h\in R}\ (-1)^{[h]_{10}[i]_{10}/2^{n-1}}\ [e^{J\cdot 0\cdot (2\pi\ [i]_{10}/l)} + e^{J\cdot 1\cdot (2\pi\ [i]_{10}/l)} +$$
$$.. + e^{J\cdot (l-1)\ (2\pi\ [i]_{10}/l)}]M_h$$

The term:

$$e^{J\cdot 0\cdot (2\pi\ [i]_{10}/l)} + e^{J\cdot 1\cdot (2\pi\ [i]_{10}/l)} + .. + e^{J\cdot (l-1)\cdot (2\pi\ [i]_{10}/l)}$$

is the summation of the l roots of order l of the unity, unless i is a multiple of l. The summation of the roots of a given order of the unity is always null. **[00046]** Therefore, in the first column of G, only those cells whose row label is $|i>$ with i multiple of l are non-null. This means that by applying G to vector $|0..0>$, measuring the result and encoding back into their binary values the first n base vectors of dimension 2 in the resulting tensor product, only strings i, such that $i=m*l$ for some integer m are obtained. This means: $i\equiv 0 \bmod l$.

DECODER:

[00047] As for Simon's algorithm, the quantum block is repeated several times to build a collection of vectors $|i\rangle$ such that $i \equiv 0 \bmod l$. By putting these equations in a system and solving it, the value of l is obtained. Since $l=2^n/r$, $r=2^n/l$ is calculated.

[00048] The number of vectors necessary for calculating r depends on the technique used for solving the system of equations. In general, it is necessary to repeat the quantum block for a number of times that increases according to a polynomial rule with n.

[00049] If 2^n is not a multiple of r, then $l_h=[2^n/r]$, for some values of h, and $l_h=[2^n/r]+1$, for some other values of h. The term

$$e^{J\cdot 0\ (2\pi\ [i]_{10}/l_h)} + e^{J\cdot 1(2\pi\ [i]_{10}/l_h)} + \ldots + e^{J\cdot(l_h-1)\ (2\pi\ [i]_{10}/\ l_h)}$$

is not exactly 0 when i is not a multiple of l_h, although it approximates 0. Therefore, all possible strings may be found as a result of measurement, but strings i that do not represent a multiple of $2^n/r$ are less likely to be found. To decrease this probability (and increase the probability of $2^n/r$-multiples), $2n$ input bits are used for encoding f. This means that more roots of the unity are involved and thus a better approximation is reached.

[00050] According to classic approaches, it is evident that the number of qubits (quantum bits) of a quantum algorithm may be a critical parameter that noticeably limits the calculation speed. In fact, by referring to the scheme of FIGURE 6, it may be noticed

that by adding only one qubit, the size of matrices with respect to the previous configuration is doubled and the number of multiplications to be performed grows exponentially. This fact limits the possibility of practically using the Shor's and Simon's algorithms when the number n of qubits is relatively large.

Summary of the Invention

[00051] In view of the foregoing background, an object of the present invention is to provide a faster method for performing Simon's or Shor's quantum algorithms.

[00052] Differently from known methods, and according to the method of the present invention, the superposition matrix operator is not generated and row-columns products are not carried out because only the indices of the non-null components of vectors generated by the superposition operation are calculated. In doing so, only vectors of 2^n components need to be calculated and stored, resulting in a significant savings in terms of computational complexity, time and memory space.

[00053] More precisely, an object of the invention is to provide a method for performing a Simon's or Shor's quantum algorithm over a certain function $f(x)$ encoded with a certain number n of qubits, that comprises carrying out a superposition operation over a set of input vectors and generating a superposition vector; carrying out an entanglement operation and generating a corresponding entanglement vector; and carrying out an interference operation and generating a corresponding output vector.

[00054] The method of the invention carries out the superposition operation in a comparably very fast

manner because it generates the superposition vector by identifying only the non-null components thereof by calculating, as a function of the number n of qubits, the value $1/2^{n/2}$ of the non-null components of the superposition vector, and by calculating the indices of these components according to an arithmetic succession, the seed of which is 1 and the common difference is 2^n. According to a preferred embodiment, similar operations are performed also for simplifying the entanglement operation.

[00055] The method of the invention may be implemented in a corresponding quantum gate that comprises a superposition subsystem carrying out a superposition operation according to a Simon's or Shor's quantum algorithm over a set of input vectors, and outputs a superposition vector. An entanglement subsystem processes components of the superposition vector, and outputs a corresponding entanglement vector. An interference subsystem processes components of the entanglement vector, and generates a corresponding output vector.

[00056] A characterizing feature of the quantum gate of the invention is based upon the superposition subsystem comprising a circuit generating a first bit string representing the value $1/2^{n/2}$ of the non-null components of the superposition vector and other 2^n bit-strings each representing a respective index of the 2^n non-null components of the superposition vector. A memory buffer stores the strings representing the value $1/2^{n/2}$ and the indices.

Brief Description of the Drawings

[00057] The various aspects and advantages of the invention will become even more evident through a detailed description referring to the attached drawings, wherein:

[00058] FIGURE 1 is a block diagram of quantum algorithms according to the prior art;

[00059] FIGURE 2 is a block diagram of an encoder according to the prior art;

[00060] FIGURE 3 is a general structure of the Quantum Block as show in FIGURE 1;

[00061] FIGURE 4 is a circuit for a Deutsch-Jozsa's quantum gate according to the prior art;

[00062] FIGURE 5a shows an example of tensor product transformation according to the prior art;

[00063] FIGURE 5b shows an example of dot product transformation according to the prior art;

[00064] FIGURE 5c shows the identity transformation according to the prior art;

[00065] FIGURE 5d shows an example of a propagation rule according to the prior art;

[00066] FIGURE 5e shows an example of an iteration rule according to the prior art;

[00067] FIGURE 5f explains the input/output tensor rule according to the prior art;

[00068] FIGURE 6 illustrates the Shor's quantum algorithm according to the prior art;

[00069] FIGURE 7 illustrates the Shor's quantum gate according to the present invention; and

[00070] FIGURE 8 illustrates the Simon's quantum gate according to the present invention.

Detailed Description of the Preferred Embodiments

[00071] Both for the Shor's algorithm as well as for the Simon's algorithm, the superposition block is $"H \otimes "I$. This implies that the first n qubits of the input vector are to be multiplied for $"H$ and the second n qubits are to be multiplied for $"I$.

[00072] To better illustrate how, according to the invention, the superposition operation may be greatly simplified, consider the following example with n=3. The first 3 qubits of the vector P generated by the superposition operator are given by:

$$\mathbf{H}|0\rangle \otimes \mathbf{H}|0\rangle \otimes \mathbf{H}|0\rangle$$

Neglecting the constant $1/2^{3/2}$ (n=3), this tensor product is:

$$\begin{bmatrix}1\\1\end{bmatrix} \otimes \begin{bmatrix}1\\1\end{bmatrix} \otimes \begin{bmatrix}1\\1\end{bmatrix} = [1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1]^T$$

In general the vector P obtained with the superposition operation is

$$P = [p_1 \quad p_2 \quad \cdots \quad p_{2^n}]$$

wherein $p_i = 1/2^{n/2}$.

[00073] For the considered example with n=3, this vector P is

$$P = \frac{1}{2^{3/2}} \cdot [1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1]^T \otimes [1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0]^T$$

In the general case (any value of n) the components p_i of the vector P obtained with the superposition operation according to the Shor's or Simon's algorithm are:

$$p_i = \begin{cases} \frac{1}{2^{n/2}} & \text{if } i = 1 + 2^n(j-1) \\ 0 & \text{otherwise} \end{cases} \qquad (6)$$

wherein $j = 1 \ldots 2^n$, $i = 1 \ldots 2^{2n}$.

[00074] The components of the superposition vector may assume only two values, only 2^n of which are non-null and must be stored. An advantage with respect to the known methods is that it is no longer necessary to store the whole superposition operator ${}^nH \otimes {}^nI$, which has $2^{2n} \times 2^{2n}$ components, and performing row-columns products, but it is possible instead to store only the positions of the 2^n non-null components of the superposition vector. The non-null value of these components is $1/2^{n/2}$ and may be calculated immediately by knowing the number n of qubits.

[00075] According to the method of the invention, the superposition vector P is not calculated carrying out 2^{2n} row-column products, as in the known methods, but is determined simply by calculating the positions of the 2^n non-null components, the value of which is $1/2^{n/2}$. These indices may be easily calculated, for instance in the form of bit strings with a complex programmable logic device (CPLD), by simply providing the number n of qubits. According to a preferred embodiment of the invention the entanglement operation is also made less complex from a computational point of view than in the

known methods.

[00076] It has been noticed that in all quantum algorithms, the entanglement matrix U_F is a diagonal block matrix, the structure of which is strictly tied to the binary representation of the function f. The diagonal matrix blocks are chosen from the set formed by the identity matrix I, the complement matrix C and matrices obtained as tensor products thereof.

[00077] For instance, for the Shor's algorithm, with n=2, the diagonal matrix blocks of the matrix U_F are chosen from the set composed of

$$I \otimes I,\ I \otimes C,\ C \otimes I,\ C \otimes C$$

each having a size $2^2 \times 2^2$. These matrix blocks may be immediately obtained from TABLE 1 noticing that, if the value 00 corresponds to a certain vector (x_0, x_1), then the matrix block on the diagonal of U_F in correspondence of the row $|x_0,x_1\rangle$ is $I \otimes I$. Similarly, when $f(x_0, x_1)$ is equal to 01, 10 or 11, the matrix block on the diagonal of U_F in correspondence of the row $|x_0,x_1\rangle$ is respectively equal to $I \otimes C$, $C \otimes I$, $C \otimes C$. This rule is resumed in the following table:

TABLE 20

$F(x_0, x_1) = 00$	$F(x_0, x_1) = 01$	$f(x_0, x_1) = 10$	$f(x_0, x_1) = 11$
$I \otimes I$	$I \otimes C$	$C \otimes I$	$C \otimes C$

In general, when this bit-string is known

$$f(x_0, \ldots, x_{n-1}) = (z_0, \ldots, z_{m-1})$$

it is possible to recognize immediately that the matrix block on the main diagonal (up-left to down-right) of U_F in correspondence of the row $|x_0, \cdots, x_{n-1}\rangle$ is:

$$(z_0 \cdot C + (1 - z_0) \cdot I) \otimes \cdots \otimes (z_{m-1} \cdot C + (1 - z_{m-1}) \cdot I)$$

This straightforward rule may be easily verified and it holds for all quantum algorithms. It allows a quick calculation of the non-null 2^n matrix blocks of U_F, each having $2^n \times 2^n$ components.

[00078] By knowing the non-null matrix blocks of the matrix U_F, it is possible to quickly calculate the vector A obtained with the entanglement operation, recalling that only a component per each row and per each column of the matrix U_F is non-null. According to a preferred embodiment of the method of the invention it is possible to calculate immediately the components a_k of the entanglement vector A without calculating any product.

[00079] In more detail, it has been observed that only the first row of each $2^n \times 2^n$ matrix block of the entanglement operator U_F gives a non-null result, because the non-null components of the superposition vector P of Shor's and Simon's algorithms are given by eq. (6). As a consequence, it is even possible to calculate the components a_k of the entanglement vector A by directly using the following formula

$$a_k = \begin{cases} \frac{1}{2^{n/2}} & \text{if } k = f(j) + 1 + 2^n(j-1) \\ 0 & \text{otherwise} \end{cases} \quad (7)$$

wherein $j=1...2^n$, $k=1...2^{2n}$.

[00080] The result is a vector the components of which are null or assume a same non-null value that depends only on the number n of qubits. As for the superposition vector, also in this case there are only 2^n non-null components having the same value, thus the entanglement vector may be determined by the indices of the 2^n non-null components.

[00081] Therefore, the entanglement operation, as well as the superposition operation, may be carried out with a circuit that generates bit-strings that represent these indices and a memory buffer in which to store these strings. It is much more difficult to perform the interference operation of the Shor's algorithm. In fact, differently from the entanglement operation, the vector components are not composed exclusively by two values. Moreover the presence of tensor products, the number of which grows rapidly with n, forms a critical computational problem.

[00082] To simplify the input-output relations, certain particular properties of the matrix $QFT_n \otimes^n I$ have been considered. Differently from other quantum algorithms, the interference operation in the Shor's algorithm is performed through a quantum Fourier transformation QFT. As all quantum operators, the QFT operator is a unary operator acting on complex vectors of the Hilbert space. It transforms each input vector in a superposition of base vectors of the same amplitude, but with a phase shift.

[00083] The interference matrix $QFT_n \otimes^n I$ has only a few non-null components. It has $2^n \cdot (2^n - 1)$ zeroes in each column. As a consequence, it is not necessary to

perform all multiplications because many of them give a null result.

[00084] Since b_h is the components of the vector B generated with the interference operation, the real part and the imaginary part of b_h are given by the following equations:

$$\mathrm{Re}[b_h] = \sum_{j=1}^{2^n} a_{(h \bmod 2^n)+1+2^n(j-1)} \cos\left(2\pi \frac{(j-1)\cdot \mathrm{int}[(h-1)/2^n]}{2^n}\right)$$

$$\mathrm{Im}[b_h] = \sum_{j=1}^{2^n} a_{(h \bmod 2^n)+1+2^n(j-1)} \sin\left(2\pi \frac{(j-1)\cdot \mathrm{int}[(h-1)/2^n]}{2^n}\right)$$

since int[.] is the function that generates the integer part of its argument and since h=1..2^{2n}.

[00085] Therefore, according to the improved method of the invention, each component is calculated simply by performing a summation of at most 2^n cosine (or sine) functions and multiplying this summation for $1/2^{n/2}$, because the non-null components of the entanglement vector A are equal to $1/2^{n/2}$.

[00086] The improvement with respect to the known method is evident, because they contemplate the step of calculating the real and imaginary parts of the components of the output vector by performing a summation of 2^{2n} products, thus with a significantly greater computational complexity.